

# Fast computation of Tukey trimmed regions in dimension $p > 2$

Xiaohui Liu<sup>a,b</sup>      Karl Mosler<sup>c</sup>      Pavlo Mozharovskyi<sup>c</sup>

<sup>a</sup>School of Statistics, Jiangxi University of Finance and Economics  
Nanchang, Jiangxi 330013, China

<sup>b</sup>Research Center of Applied Statistics, Jiangxi University of Finance and Economics  
Nanchang, Jiangxi 330013, China

<sup>c</sup>Institute of Econometrics and Statistics, University of Cologne  
Albertus-Magnus-Platz, 50923 Cologne, Germany

December 17, 2014

## Abstract

Given data in  $\mathbb{R}^p$ , a Tukey  $\tau$ -trimmed region is the set of all points that have at least Tukey depth  $\tau$  in the data. As they are visual, affine equivariant and robust, Tukey trimmed regions are useful tools in nonparametric multivariate analysis. While these regions are easily defined and can be interpreted as  $p$ -variate quantiles, their practical application is impeded by the lack of efficient computational procedures in dimension  $p > 2$ . We construct two algorithms that exploit certain combinatorial properties of the regions. Both calculate the exact region. They run much faster and require substantially less RAM than existing ones.

**Keywords:** Tukey depth; Location depth; Halfspace depth; Depth contours; Depth regions; Computational geometry; Combinatorial algorithm; Exact algorithm; Efficient implementation.

**2000 MSC Codes:** 62F10; 62F40; 62F35.

## 1 Introduction

To describe the centrality of a point in a multivariate set of data, Tukey (1975) made a celebrated proposal: Given data  $x_1, \dots, x_n$  in  $\mathbb{R}^p$  and an

additional point  $x$ ,

$$d(x|P_{\mathcal{X}}) = \inf_{u \in \mathcal{S}^{p-1}} \frac{1}{n} \# \{i \in \{1, \dots, n\} : u^{\top} x_i \leq u^{\top} x\} \quad (1)$$

measures how central  $x$  is situated with respect to the data. Here  $\mathcal{S}^{p-1} = \{v \in \mathbb{R}^p : \|v\| = 1\}$  denotes the unit sphere in  $\mathbb{R}^p$ ,  $\#(B)$  the cardinal number of a set  $B$ , and  $P_{\mathcal{X}}$  is the empirical distribution function of the data. As a function of  $x$ , (1) is referred to in the literature as *location depth* or *halfspace depth*, but also, to reflect the seminal work of Tukey, as *Tukey depth*. The Tukey depth takes its maximum at the *Tukey median*, which in general is not unique. The depth decreases when  $x$  moves on a ray originating from a Tukey median and vanishes outside the convex hull  $\text{conv}(x_1, \dots, x_n)$  of the data. Thus, the depth provides a center-outward ordering of points in  $\mathbb{R}^p$ .

The Tukey depth is invariant against a simultaneous affine transformation of  $x$  and the data. Moreover, and most important, its value does not change as long as any of the points  $x, x_1, \dots, x_n$  is moved without crossing an observation hyperplane. By an *observation hyperplane* we mean a hyperplane that is generated by  $p - 1$  elements of  $\{x, x_1, \dots, x_n\}$ . By the last property, the Tukey depth is rather robust against outlying observations; see Donoho (1982).

A *Tukey  $\tau$ -trimmed region* is the set of points in  $\mathbb{R}^p$  that have at least Tukey depth  $\tau$ . For  $\tau > \frac{1}{n}$  it is a closed convex polyhedron, included in  $\text{conv}(x_1, \dots, x_n)$ , hence compact. Tukey regions are nested, shrinking with increasing  $\tau$ . An empirical distribution is fully characterized by its Tukey regions (Struyf & Rousseeuw, 1999).

Similarly, the centrality of a point  $x$  regarding a probability distribution  $P$  in  $\mathbb{R}^p$  is measured by the *population version* of the Tukey depth as follows:

$$d(x|P) = \inf_{u \in \mathcal{S}^{p-1}} P[\{z \in \mathbb{R}^p : u^{\top} x \leq u^{\top} z\}] \quad (2)$$

Note that (1) is a special case of (2) with  $P_{\mathcal{X}}$ . The population version of Tukey regions determines the underlying distribution if either the distribution is discrete (Koshevoy, 2002) or the regions' boundaries are smooth (Kong & Zuo, 2010).

In multivariate analysis, a broad nonparametric methodology can be developed that is based on Tukey depth and Tukey regions, in particular, multivariate procedures of signs and ranks, order statistics, quantiles, and measures of outlyingness and risk. As the Tukey depth is affine invariant and robust, so is any inference based on it. For such procedures, see for example Yeh & Singh (1997), Serfling (2006), Mozharovskiy *et al.* (2014), and references therein. Many other depth notions have been proposed and used in

the literature for descriptive as well as inferential procedures. Among them are the simplicial depth (Liu, 1990), the zonoid depth (Koshevoy & Mosler, 1997), and the projection depth (Liu, 1992; Zuo, 2003). For recent surveys, see Mosler (2013) and (Van Bever, 2013, Ch. 1). General definitions of a depth function can be found in Zuo & Serfling (2000) and Dyckerhoff (2002).

The various notions of data depth differ in their theoretical properties: regarding invariance and robustness, convergence to their population version, and whether they fully characterize an underlying probability distribution. By this, from an applied view, different depths fit to different applications. However, most decisive in possible applications is, whether the depth can be efficiently computed for real sample sizes and in higher dimensions. To calculate the Tukey depth, feasible algorithms have been developed by Ruts & Rousseeuw (1996) for dimension  $p = 2$ , Rousseeuw & Struyf (1998) for  $p = 3$ , and most recently by Dyckerhoff & Mozharovskiy (2014) and Liu (2014) for general  $p$ ; see also Miller *et al.* (2003).

To compute a Tukey region of given data  $\{x_1, \dots, x_n\}$  appears as an even more challenging combinatorial task, as it involves a very large number of observation hyperplanes to be possibly inspected. In dimension two the task has been solved by use of a circular sequence, which enumerates all intersections of observation hyperplanes (Ruts & Rousseeuw, 1996). Kong & Mizera (2012) demonstrate that a Tukey  $\tau$ -trimmed region is bordered by hyperplanes  $\pi_{\text{KM}}(\tau, u)$  which correspond to  $\tau$ -quantiles of projections in direction of their normals  $u$ . Moreover, the Tukey is the *infinite* intersection over all directions  $u$  of the inner halfspaces characterized by  $\pi_{\text{KM}}(\tau, u)$ .

But, as a convex polytope, a Tukey trimmed region is the intersection of a *finite* number of halfspaces. The facets of the polytope lie on the hyperplanes that border the halfspaces. Clearly, by the definition of Tukey depth (1), each of these hyperplanes must be an observation hyperplane. Consequently, the Tukey region is completely determined by a finite number of observation hyperplanes. To calculate it, a naive procedure consists in passing through all  $\binom{n}{p}$  observation hyperplanes and checking their depth. For a more efficient procedure, we have to identify a proper subset of observation hyperplanes that contain the facets.

Hallin *et al.* (2010), hereafter HPS, point out a direct connection between Tukey regions and multivariate regression quantiles. Each such quantile is a set of hyperplanes, in general non-unique, that may include a facet of the Tukey region. Hallin *et al.* (2010) show that those directions giving the same hyperplane form a polyhedral cone, and that a finite number of these cones fills the  $\mathbb{R}^p$ . Each cone can be represented by the direction vectors generating its edges, which, again, is a finite number. HPS provide an algorithm calculating Tukey regions in dimension  $p > 2$  via quantile regression

and parametric programming. To guarantee all cones to be addressed, a breadth-first search is used. For details see Paindaveine & Šiman (2012a,b, 2011). However this procedure is rather inefficient and slow.

Here two new algorithms are presented that calculate a Tukey  $\tau$ -trimmed exactly in arbitrary dimension  $p > 2$ . In doing this we identify certain combinatorial properties of Tukey regions and exploit them in order to substantially reduce the computational load. Consequently the new algorithms run much faster and require much less RAM than the one by HPS.

The HPS procedure is inefficient for two reasons. First, it appears that their method yields a great number of *redundant* direction vectors, which result in no facet of the trimmed region. All these direction vectors are considered and used to calculate regions. Given  $\tau$ , the HPS procedure calculates  $p$  regions instead of one by breadth-first; but many of these regions have depth  $\neq \tau$  (see Paindaveine & Šiman (2011), remark after Theorem 4.2). Second, the *cone-by-cone* search strategy is both RAM- and time-consuming. This is because, in their procedures, each cone is characterized by its facets and vertices, and facets are identified by  $p$ -variate vectors. A rather large RAM is required to store these identifiers with sufficient precision.

In the sequel we construct two new algorithms for exactly computing a Tukey trimmed region, that are more efficient. The first algorithm is naive, and serves mainly as a benchmark for the second one. In searching for facets' candidates, it simply passes through all combinations of  $p - 1$  observations as the case may be. No memory-consuming structure has to be created, and the computation time is independent of  $\tau$ . The second algorithm uses a breadth-first search like the one of HPS. However, instead of *cone-by-cone*, it searches the direction vectors *ridge-by-ridge*, where a *ridge* corresponds to a set of  $p - 1$  observations in  $\mathbb{R}^p$ . We store each ridge by the subscripts of its  $p - 1$  corresponding observations and use some novel tricks to substantially save both RAM and computational time.

Our approach exploits the connection between Tukey trimmed region and quantile regions pointed out by Kong & Mizera (2012), *viz.* that the Tukey  $\tau$ -region is the intersection of directional  $\tau$ -quantile halfspaces, taken over all directions of the unit sphere. While it has been argued in the literature (Hallin *et al.*, 2010, see page 652) that it is impossible to found a feasible procedure on this connection, we show that this is not the case. Specifically we demonstrate that a finite number of Kong and Mizera's  $\tau$ -quantile halfspaces yields the Tukey  $\tau$ -region, and we characterize these halfspaces in a way which allows for calculating them efficiently.

The new algorithms offer an efficient approach to calculate Tukey regions in arbitrary dimension and, by this, enable the use of statistical methodology based on these set-valued statistics. A simulation study as well as real data

calculations (up to dimension 9) are provided below to illustrate the performance of the algorithms. Our procedures have been implemented in C++ and visualized in **R**. The complete code can be obtained from the authors.

The rest of this paper is organized as follows. Section 2 investigates the connection between Tukey trimmed regions and quantile regions. Section 3 provides the algorithms. Section 4 illustrates the performance of the proposed algorithms with real and simulated data. Section 5 concludes.

## 2 Constructing trimmed regions as quantile envelopes

Consider an empirical distribution  $P_{\mathcal{X}}$  on a set  $\mathcal{X} = \{x_1, \dots, x_n\}$  of data in  $\mathbb{R}^p$ . Its *Tukey  $\tau$ -trimmed region* is the upper level set of the Tukey depth (1) at some level  $\tau$ ,

$$\mathcal{D}(\tau) = \{x \in \mathbb{R}^p : d(x, P_{\mathcal{X}}) \geq \tau\}. \quad (3)$$

To avoid unbounded as well as empty regions, (3) is restricted to  $\tau \in [\frac{1}{n}, \sup_{x \in \mathbb{R}^p} d(x, P_{\mathcal{X}})]$ . As Kong & Mizera (2012) have pointed out, the concept of Tukey regions is closely connected with that of projection quantiles. For any  $u$  in the unit sphere  $\mathcal{S}^{d-1}$ , define a *projection quantile* in the sense of Kong & Mizera (2012) as

$$\begin{aligned} q_{\text{KM}}(\tau, u) &= q_1(\tau, u)u, \quad \text{where} \\ q_1(\tau, u) &= \min\{t : F_{u^\top \mathcal{X}}(t) \geq \tau\}. \end{aligned} \quad (4)$$

Here  $F_{u^\top \mathcal{X}}$  is the empirical distribution function of  $\mathcal{X}$  projected in direction  $u$ ,  $q_1(\tau, u)$  is the usual univariate quantile of  $F_{u^\top \mathcal{X}}$ . The projection quantile  $q_{\text{KM}}(\tau, u)$  is the vector pointing in direction  $u$  having length  $q_1(\tau, u)$ . Further, define the halfspace

$$\mathcal{H}_{\text{KM}}(\tau, u) = \{z \in \mathbb{R}^p : u^\top z \geq u^\top q_{\text{KM}}(\tau, u)\}. \quad (5)$$

It is bounded away from the origin at distance  $q_1(\tau, u)$  by the hyperplane with normal  $u$ . Recall that an observation hyperplane is generated by any  $p$  points out of the data  $x_1, \dots, x_n$ . The data is *in general position* if each observation hyperplane contains exactly  $p$  data points. Kong & Mizera (2012) demonstrate that the Tukey region equals the *projection quantile envelope*, that is

$$\mathcal{D}(\tau) = \bigcap_{u \in \mathcal{S}^{p-1}} \mathcal{H}_{\text{KM}}(\tau, u), \quad (6)$$

provided the data is in general position. Obviously, when extending definition (5) to all  $u \neq 0$ , it holds  $\mathcal{H}_{\text{KM}}(\tau, \lambda u) = \mathcal{H}_{\text{KM}}(\tau, u)$  for any  $\lambda > 0$ . Therefore, (6) is equivalent to

$$\mathcal{D}(\tau) = \bigcap_{u \in \mathbb{R}^p \setminus \{0\}} \mathcal{H}_{\text{KM}}(\tau, u). \quad (7)$$

HPS argues in several papers (Hallin *et al.*, 2010; Paindaveine & Šíman, 2011) that equations (6) and (7) are of no use in calculating the trimmed region. Contrary to this, we will show that the infinite intersection on their right side can be reduced to a finite one. We will characterize a finite subset of relevant directions, which can be worked through in an algorithm. Before formulating our main theoretical result, we have to introduce certain cones of directions and prove two Lemmas about them.

Given  $u \in \mathbb{R}^p \setminus \{0\}$ , there exists a permutation  $\pi_u$  of  $\{1, \dots, n\}$ , possibly non-unique, such that

$$u^\top x_{\pi_u(1)} \leq u^\top x_{\pi_u(2)} \leq \dots \leq u^\top x_{\pi_u(n)}. \quad (8)$$

Note that, if the data are in general position, at least  $n-1$  of these inequalities are strict. Let  $k_\tau = \lfloor n\tau \rfloor$ . For any  $u \neq 0$  define a *direction cone*

$$C(u, \tau) = \{v \in \mathbb{R}^p : v^\top x_{\pi_u(s)} \leq v^\top x_{\pi_u(t)} \text{ if } \pi_u(s) \leq \pi_u(k_\tau) < \pi_u(t)\}. \quad (9)$$

**Lemma 1** *It holds*

- (i) 
$$q_{\text{KM}}(\tau, u) = u^\top x_{\pi_u(k_\tau)} u. \quad (10)$$
- (ii) *There exist  $u_1, \dots, u_{\ell_\tau}$  so that each  $C(u_k, \tau)$  has nonempty interior,  $\bigcup_{k=1}^{\ell_\tau} C(u_k, \tau) = \mathbb{R}^p$ , and every two adjacent cones share at most one facet.*
- (iii) *The function  $u \mapsto q_{\text{KM}}(\tau, u)$  is piece-wise linear on  $\mathbb{R}^p \setminus \{0\}$ .*

**Proof:** Equation (10) is obvious.  $C(u, \tau)$  is a convex closed polyhedral cone with vertex at the origin, and each facet of the cone corresponds to a non-redundant inequality in its definition. (Note that many of the  $k(n-k)$  inequality restrictions in (9) may be redundant.)  $C(u, \tau)$  contains  $u$ , hence is non-empty, and the union of all  $C(u, \tau), u \in S^{d-1}$  is the whole space  $\mathbb{R}^p$ . As there exist only finitely many permutations, the number of *different* direction cones  $C(u, \tau)$  is finite. If two cones are *adjacent* (that is, have nonempty intersection), they share at most one facet since they are convex. Finally,  $x_{\pi_u(k_\tau)}$  is constant on each of finitely many direction cones, hence the

projection quantile  $q_{\text{KM}}(\tau, u) = u^\top x_{\pi_u(k_\tau)} u$  depends piece-wise linearly on its direction  $u$ .  $\square$

Next we consider convex subcones of  $C(u, \tau)$ ,

$$C_j(u, \tau) = \{v \in \mathbb{R}^p : v^\top x_{\pi_u(s)} \leq v^\top x_{\pi_u(j)} \text{ for } s = 1, \dots, j-1, \quad (11)$$

$$v^\top x_{\pi_u(j)} \leq v^\top x_{\pi_u(t)} \text{ for } t = j+1, \dots, n\},$$

$j = 1, \dots, k_\tau$ . Clearly,  $C_j(u, \tau) \subset C(u, \tau)$  for each  $j$ , and it holds  $C(u, \tau) = \bigcup_{j=1}^{k_\tau} C_j(u, \tau)$ . Let  $W_j(u, \tau) = \{w_{j1}, \dots, w_{jm_j}\}$  denote the set of unit vectors that generate an edge of  $C_j(u, \tau)$ , and let  $W(u, \tau) = \bigcup_{j=1}^{k_\tau} W_j(u, \tau)$ .

**Lemma 2** *It holds*

$$\bigcap_{v \in W(u, \tau)} \mathcal{H}_{\text{KM}}(\tau, v) = \bigcap_{v \in C(u, \tau)} \mathcal{H}_{\text{KM}}(\tau, v). \quad (12)$$

**Proof:** If  $v_{\pi_u(k_\tau), j}^\top x_{\pi_u(k_\tau)} \leq v_{\pi_u(k_\tau), j}^\top z$  for all  $j = 1, \dots, m_{\pi_u(k_\tau)}$ , by the convexity of  $C_j(u, \tau)$  for any  $\lambda_i > 0$  we obtain  $(\sum_{j=1}^{m_{\pi_u(k_\tau)}} \lambda_j v_{\pi_u(k_\tau), j})^\top x_{\pi_u(k_\tau)} \leq (\sum_{j=1}^{m_{\pi_u(k_\tau)}} \lambda_j v_{\pi_u(k_\tau), j})^\top z$ , hence

$$\bigcap_{i=1}^{m_j} \mathcal{H}_{\text{KM}}(\tau, w_{ji}) \subset \mathcal{H}_{\text{KM}}(\tau, \sum_{i=1}^{m_j} \lambda_i w_{ji}).$$

It follows

$$\bigcap_{v \in W_j(u, \tau)} \mathcal{H}_{\text{KM}}(\tau, w_v) \subset \bigcap_{v \in C_j(u, \tau)} \mathcal{H}_{\text{KM}}(\tau, v), \quad (13)$$

and, consequently, as the opposite inclusion is obvious, equality in (13). By going to the union of index sets on both sides, (12) is obtained.  $\square$

Note that, *not for every*  $v \in W(u, \tau)$  the halfspace  $\mathcal{H}_{\text{KM}}(\tau, v)$  cuts off exactly  $\lfloor n\tau \rfloor - 1$  observations, because  $v$  may be normal to some  $x_s - x_t$  with  $s \neq t$  and  $s, t \in \{1, 2, \dots, k_\tau\}$ . In the following, we will show that those directions  $v$  can be identified, and (12) still holds with a smaller  $W(u, \tau)$  that has been purified of them. In contrast to this, the algorithm of HPS calculates *all* hyperplanes that are *normal to these directions* by quantile regression.

**Theorem 1** *For any  $\frac{1}{n} \leq \tau \leq \tau^*$ , there exists a finite set of critical directions,  $\mathcal{U}_\tau = \{u_{\tau 1}, \dots, u_{\tau m_\tau}\} \subset \mathcal{S}^{p-1}$ , such that*

$$\mathcal{D}(\tau) = \bigcap_{j=1}^{m_\tau} \mathcal{H}_{\text{KM}}(\tau, u_{\tau j}).$$

*Each  $u_{\tau j}$ ,  $j = 1, \dots, m_\tau$ , is a unit direction vector that*

- (i) generates an edge of a cone in  $\{C_j(u, \tau) : u \in S^{d-1}, j = 1, \dots, k_\tau\}$ ,
- (ii) is orthogonal to a hyperplane through  $p$  observations that cuts off exactly  $\lfloor n\tau \rfloor - 1$  observations.

Theorem 1 indicates how in computing  $\mathcal{D}(\tau)$  we can restrict ourselves to a finite number of enveloping halfspaces  $\mathcal{H}_{\text{KM}}(\tau, u)$ . The normals of these halfspaces are characterized in a way which gives path to an efficient calculation of the Tukey region. Part (i) clarifies that the relevant directions correspond to edges of certain cones, and Part (ii) provides a rule to select them. Further, Theorem 1 tells us the connection between  $\mathcal{U}_\tau$  and cones of type  $\mathcal{C}(u, \tau)$  which are cut by hyperplanes normal to some  $x_i - x_k$  and passing through the origin. (Compare the results of Hallin *et al.* (2010) and Theorem 4.2 in Paindaveine & Šiman (2011).) Here we identify such hyperplanes simply by subscripts, say,  $(\pi_u(s), \pi_u(t))$ , where  $s \in \{1, \dots, k_\tau\}$  and  $t \in \{k_\tau + 1, \dots, n\}$ . Furthermore, from the proof of Theorem 1, we will see why, if  $p > 2$ , not all direction vectors lying in the vertices of  $C(u, \tau)$  are critical directions at level  $\tau$ .

**Proof of Theorem 1:** Though our algorithms are designed for  $p \geq 3$ , we start with the case  $p = 2$ . Consider some  $1 \leq i < k \leq n$  such that the line through  $x_i$  and  $x_k$  cuts off exactly  $\lfloor n\tau \rfloor - 1$  points from the data. Let  $w = w_{ik}$  be the unit vector that is orthogonal to  $x_i - x_k$  and points into the inner part of  $H_{ik} := \{z \in \mathbb{R}^p : w^\top z \geq w^\top x_i = w^\top x_k\}$ . Let us look at the cone  $C(w, \tau)$  and its subcones  $C_j(w, \tau)$  by cases.

- (a) If  $C_j(w, \tau) = \emptyset$  for all  $j = 1, \dots, k_\tau - 1$ , then  $C(w, \tau) = C_{k_\tau}(w, \tau)$ , and any non-redundant facet of  $C(w, \tau)$  must be normal to some  $x_s - x_t$  with  $s \in \{\pi_u(1), \dots, \pi_u(k_\tau)\}$  and  $t \in \{\pi_u(k_\tau + 1), \dots, \pi_u(n)\}$ .
- (b) If just two of the subcones are nonempty, say, having indices 1 and 2, we obtain  $C(w, \tau) = C_1(w, \tau) \cup C_2(w, \tau)$ . In this case, the two cones are adjacent and share one vertex; hence  $W(w, \tau)$  contains three direction vectors. The subcone  $C_1(w, \tau)$  may be described by a triple of index sets,  $(\{\dots, 3\}, \{2\}, \{1, 4, \dots\})$ , and  $C_2(w, \tau)$  by  $(\{\dots, 2\}, \{3\}, \{1, 4, \dots\})$ , where  $\pi_u(1) = 2$  and  $\pi_u(2) = 3$ . Let  $u_{12}$  and  $u_{23}$  span the edges of  $C_1$ , and  $u_{23}$  and  $u_{34}$  those of  $C_2$ . For convenience, notate  $u_{ik}$  by its angle coordinate  $\alpha_{ik}$ , and assume that  $\alpha_{12} \geq \alpha_{23} \geq \alpha_{34}$ . It holds that  $u_{23} = \lambda_1 u_{12} + \lambda_2 u_{34}$  for some  $\lambda_1, \lambda_2 > 0$ . Note that, for all  $x \in H_{12}$ ,  $u_{12}^\top x \geq u_{12}^\top x_1 = u_{12}^\top x_2$ , and for all  $x \in H_{34}$ ,  $u_{34}^\top x \geq u_{34}^\top x_3 = u_{34}^\top x_4 > u_{34}^\top x_2$ . Hence, for any  $x \in H_{12} \cap H_{34}$ , we obtain  $\lambda_1 u_{12}^\top x + \lambda_2 u_{34}^\top x > \lambda_1 u_{12}^\top x_2 + \lambda_2 u_{34}^\top x_2$ , which further implies  $u_{23}^\top x > u_{23}^\top x_2$ . That is,  $x \in H_{23}$ , and  $H_{12} \cap H_{34} \subset H_{23}$ . This proves  $H_{12} \cap H_{23} \cap H_{34} = H_{12} \cap H_{34}$ . Hence,  $u_{23}$  can be eliminated.



- (c) If there are more than two cones, say  $C_1$ ,  $C_2$  and  $C_3$ , such that  $C(w, \tau) = C_1 \cup C_2 \cup C_3$ , we can eliminate the shared facets one-by-one clockwise similarly to (b); See Figure 1 for an illustration.

Now we assume  $p \geq 3$  and take into account all cones  $C(u_k, \tau)$ ,  $k = 1, \dots, \ell_\tau$ , according to Lemma 1. Let  $r_1, \dots, r_{M_\tau}$  denote the direction vectors that generate edges of  $C(u_k, \tau)$  for any  $k$ . If two of these vectors share the same ridge (that is, intersection of two observation hyperplanes), say, generated by  $\{x_1, \dots, x_{p-1}\}$ , these two vectors lie in a two-dimensional space  $V_{p-2}^\perp$ , which is the orthogonal complement of the subspace  $V_{p-2}$  spanned by  $x_2 - x_1, x_3 - x_1, \dots, x_{p-1} - x_1$ . This in fact reduces the task to the case of  $p = 2$ : As above one can show that, if a direction vector lies in a facet normal to some  $x_s - x_t$  such that  $s, t \in \{\pi_u(1), \dots, \pi_u(k_\tau)\}$  and  $s \neq t$ , it can be eliminated from further consideration. This completes the proof of Theorem 1.  $\square$

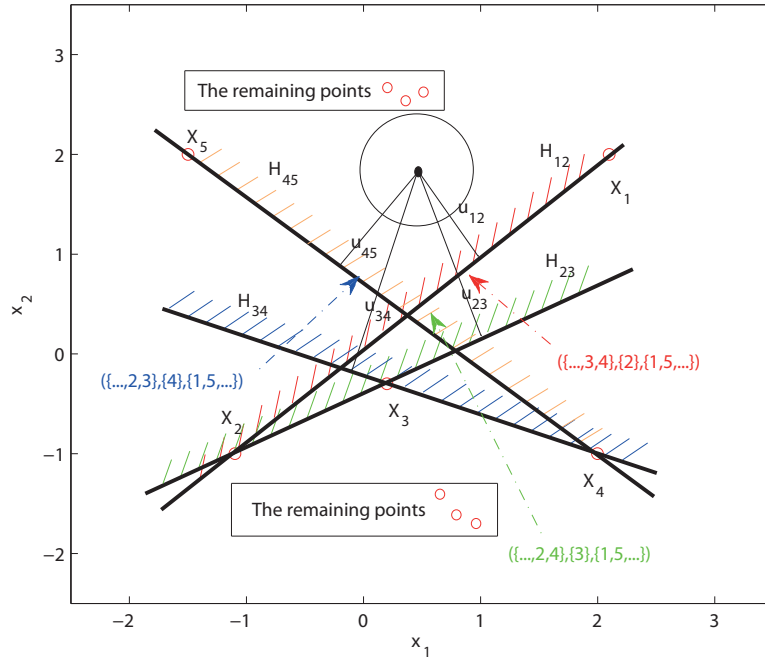


Figure 1: Shown is an illustration of the sets' intersection in Theorem 1, where  $u_{12}$  and  $u_{45}$  cut off  $\lfloor n\tau \rfloor - 1$  points, and  $u_{23}$  and  $u_{34}$  cut off only  $\lfloor n\tau \rfloor - 2$ . Clearly, we have that  $H_{12} \cap H_{23} \cap H_{34} \cap H_{45} = H_{12} \cap H_{45}$ .

Next we ask how many facets the Tukey region may have, that is how many direction vectors may be needed to compute it exactly. The following

corollary provides an upper bound of this number, and can be useful in assessing the performance of an algorithm.

**Corollary 1** *Assume that the observations  $\mathcal{X}$  are in general position. The number  $M_\tau$  of non-redundant facets of a Tukey region is bounded above by  $2\binom{n}{p-1}$  for any level  $\tau \in [\frac{1}{n}, \tau^*]$ .*

**Proof:** As stated in Theorem 1, each direction vector  $u \in \mathcal{U}_\tau$  is orthogonal to a set of  $p$  observations, some of which may have a ridge in common. For convenience, denote these shared ridges as  $\mathcal{H}_l = \{x_{i_{l,1}}, x_{i_{l,2}}, \dots, x_{i_{l,p-1}}\}$ ,  $l = 1, 2, \dots, M_0$ , where  $M_0$  is their total number. Let  $V_{l,p-2}$  denote the space spanned by  $\{x_{i_{l,2}} - x_{i_{l,1}}, x_{i_{l,3}} - x_{i_{l,1}}, \dots, x_{i_{l,p-1}} - x_{i_{l,1}}\}$ . Then, for  $p \geq 2$ ,

$$\mathcal{D}(\tau) = \bigcap_{u \in \mathcal{U}_\tau} \mathcal{H}_{\text{KM}}(\tau, u) = \bigcap_{l=1}^{M_0} \left( \bigcap_{u \in \mathcal{U}_\tau, u \perp V_{l,p-2}} \{\mathcal{H}_{\text{KM}}(\tau, u)\} \right). \quad (14)$$

If  $p = 2$ , we write  $V_{l,0} = \{0\}$ , namely, the set containing only the origin. Consider the orthogonal complement  $V_{l,p-2}^\perp$  of  $V_{l,p-2}$ , so that  $\mathbb{R}^p = V_{l,p-2}^\perp \oplus V_{l,p-2}$ . Let  $U_l^* = \mathcal{U}_\tau \cap V_{l,p-2}^\perp$ , and  $n_l = \#U_l^*$ . A given  $z \in \mathbb{R}^p$  is decomposed as  $z = z' + z''$  with  $z' \in V_{l,p-2}^\perp$  and  $z'' \in V_{l,p-2}$ ; similarly,  $x_{i_{l,1}} = x'_l + x''_l$ . Then, for each  $u \in U_l^*$ , we obtain

$$\begin{aligned} \mathcal{H}_{\text{KM}}(\tau, u) &= \{z \in \mathbb{R}^p : u^T z \geq u^T x_{i_{l,1}}\} \\ &= \{z \in \mathbb{R}^p : u^T (z' + z'') \geq u^T (x'_l + x''_l)\} \\ &= \{z \in \mathbb{R}^p : u^T z' \geq u^T x'_l\} \\ &= \mathcal{H}_l(\tau, u) \oplus V_{l,p-2}, \end{aligned}$$

where  $\mathcal{H}_l(\tau, u) = \{z \in V_{l,p-2}^\perp : u^T z' \geq u^T x'_l\}$ . Note that the boundary hyperplanes of  $\mathcal{H}_l(\tau, u)$  pass through the common point  $x'_l$ . Using this and the convexity of both  $\mathcal{D}(\tau)$  and the halfspace, it is easy to show that, when the observations are in general position, there exist two subscripts (see Figure 2 for an illustration), namely  $k_1$  and  $k_2$ , such that

$$\begin{aligned} \bigcap_{u \in \mathcal{U}_\tau, u \perp V_{l,p-2}} \mathcal{H}_{\text{KM}}(\tau, u) &= V_{l,p-2} \oplus \bigcap_{l=1}^{n_l} \mathcal{H}_l(\tau, u) \\ &= V_{l,p-2} \oplus (\mathcal{H}_{k_1}(\tau, u) \cap \mathcal{H}_{k_2}(\tau, u)) \\ &= \mathcal{H}_{\text{KM}}(\tau, u_{l,k_1}) \cap \mathcal{H}_{\text{KM}}(\tau, u_{l,k_2}). \end{aligned}$$

This, combined with (14) and the fact that  $M_0 = O(\binom{n}{p-1})$ , proves Corollary 1.  $\square$

By the convexity of the Tukey region, a critical direction vector yields at most one facet of the corresponding trimmed region. In this sense, Corollary 1 actually also provides an upper bound for the number of the *non-redundant* critical direction vectors. The Corollary will be useful in Step 4 of Algorithm 2 below.

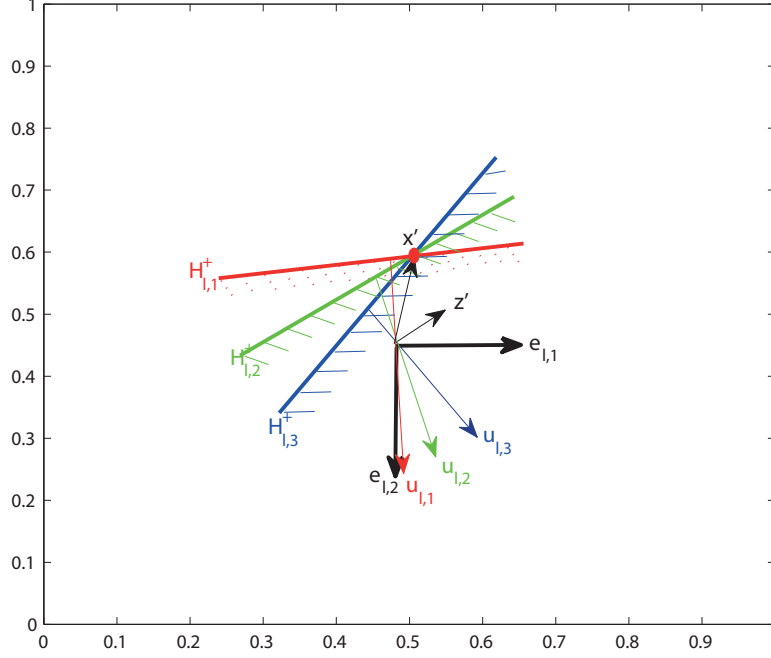


Figure 2: Shown is an illustration of the sets' intersection in Corollary 1, where  $e_{l,1}$ ,  $e_{l,2}$  denotes the canonical basis vectors of  $V_{l,p-2}^\perp$ , and  $u_{l,1}$ ,  $u_{l,2}$ ,  $u_{l,3}$  three critical direction vectors. Clearly,  $\bigcap_{k=1}^3 H_{l,k} = H_{l,1} \cap H_{l,3}$ , and  $u_{l,2}$  are redundant.

### 3 Algorithms

In this section, we present two combinatorial algorithms based on Theorem 1 for exactly computing  $\mathcal{U}_\tau$ . We assume that the observations  $\mathcal{X} = \{x_1, \dots, x_n\}$  are in general position. The first algorithm (Algorithm 1) is rather naive. It simply passes through all combinations  $\{x_{i_1}, \dots, x_{i_{p-1}}\}$  and searches for critical directions, hence facets' candidates, in the planes orthogonal to the affine subspaces they generate.

#### Algorithm 1

**Input:**  $\mathcal{X} = \{x_1, \dots, x_n\} \in \mathbb{R}^p$ ,  $2 \leq p < n < \infty$ ,  $\mathcal{X}$  in general position.

**Step 1.** Set  $\mathcal{G}_\tau = \emptyset$ .

**Step 2.** For each  $\{i_1, i_2, \dots, i_{p-1}\} \in \{1, 2, \dots, n\}$ , do:

- (a) Find all subscripts  $i_p \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{p-1}\}$  such that  $\{x_{i_1}, \dots, x_{i_{p-1}}, x_{i_p}\}$  determine a  $\tau$ -th critical direction, and store all such  $i_p$  into  $\mathcal{T}$ .
- (b) If  $\mathcal{T} = \emptyset$ , go to the next iteration, else for each  $i_p \in \mathcal{T}$  do:
  - i. Let  $[j_1, \dots, j_p]$  be the sorted  $\{i_1, \dots, i_{p-1}, i_p\}$ .
  - ii. Compute  $g = j_1 + j_2 n + j_3 n^2 + \dots + j_p n^{p-1}$ .
  - iii. Add  $g$  to  $\mathcal{V}_\tau$ .

**Output:**  $\mathcal{G}_\tau$ .

The main feature of this algorithm is: it does not take account of any space ordering like the breadth-first algorithm of Paindaveine & Šiman (2012a). Therefore it requires minimum memory and saves computation time; see e.g. Mozharovskiy (2014) for a similar discussion in computing the Tukey depth.

On the other hand, this algorithm has to fully address all combinations no matter what the value of  $\tau$  is. When  $\tau$  is small, a great deal of computation time may be wasted on checking the redundant combinations that do not determine halfspaces cutting off  $\lfloor n\tau \rfloor - 1$  points. As an improvement, the following algorithm (Algorithm 2) browses only through those combinations that determine the  $\tau$ -th direction vectors. It utilizes the breadth-first algorithm; but this searches over the combinatorial structures only, and is, by that, very fast.

## Algorithm 2

**Input:**  $\mathcal{X} = \{x_1, \dots, x_n\} \in \mathbb{R}^p$ ,  $2 < p < n < \infty$ ,  $\mathcal{X}$  in general position.

**Step 1.** Set  $\mathcal{A} = (\text{false}_n)^{p-1}$ ,  $\mathcal{G}_\tau = \emptyset$ , and an empty queue  $\mathcal{Q}$ .

**Step 2.** Generate  $u_0 \in \mathcal{S}^{p-1}$  uniformly distributed on the unit sphere and let  $(j_1, \dots, j_n)$  be the permutation it maintains, such that  $u_0^\top x_{j_1} < \dots < u_0^\top x_{j_n}$  (which by the assumption of general position is always feasible). Do:

- (a) Check whether there exists another  $x \in \mathcal{X}$  such that the set  $\{x_{j_1}, \dots, x_{j_{p-1}}, x\}$  determines a  $\tau$ -th critical direction  $u_1$ . If not, go to **Step 2**, else proceed to the next.

- (b) Find all  $\lfloor n\tau \rfloor + p - 1$  observations  $\{x_{i_l}\}_{l=1}^{\lfloor n\tau \rfloor + p - 1}$  such that  $u_1^\top x_{i_l} \leq u_1^\top x$ .
- (c) For each  $\{k_1, k_2, \dots, k_{p-1}\} \subset \{j_1, j_2, \dots, j_{\lfloor n\tau \rfloor + p - 1}\}$ , do:
  - i. Let  $[l_1, \dots, l_{p-1}]$  be the sorted  $\{k_1, \dots, k_{p-1}\}$ .
  - ii. Set  $\mathcal{A}_{[l_1, \dots, l_{p-1}]} = \mathbf{true}$  and push  $[l_1, \dots, l_{p-1}]$  into  $\mathcal{Q}$ .

**Step 3.** Pop an  $[i_1, \dots, i_{p-1}]$  from  $\mathcal{Q}$ .

**Step 4.** Find all subscripts  $i_p \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{p-1}\}$  such that  $\{x_{i_1}, \dots, x_{i_{p-1}}, x_{i_p}\}$  determine a  $\tau$ -th critical direction vector  $u$ , and store all these subscripts in the set  $\mathcal{T}$ . Specifically,

- (a) Compute the standard orthogonal basis vectors  $e_1, e_2$  of  $V_{p-2}^\perp$ , the orthogonal complement space of  $V_{p-2}$  spanned by  $x_{i_2} - x_{i_1}, x_{i_3} - x_{i_1}, \dots, x_{i_{p-1}} - x_{i_1}$ .
- (b) For each  $k \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{p-1}\}$ , compute the polar coordinate angle  $\theta_k$  ( $\theta_k \in [-\pi, \pi)$ ) of the vector  $(x_k^\top e_1, x_k^\top e_2)^\top$ , where  $x_k = x_k - x_{i_1}$ .
- (c) For each  $k \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{p-1}\}$ , do:
  - i. Compute  $\eta_1 = \#\{l : \theta_l \in (\theta_k, \theta_k + \pi), l \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{p-1}, k\}\}$ , and  $\eta_2 = \#\{l : \theta_l \in [-\pi, \theta_k) \cup (\theta_k + \pi, \pi), l \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{p-1}, k\}\}$ .
  - ii. If either  $\eta_1 = \lfloor n\tau \rfloor - 1$  or  $\eta_2 = \lfloor n\tau \rfloor - 1$ , push  $k$  into  $\mathcal{T}$ .

**Step 5.** If  $\mathcal{T} = \emptyset$ , go to the next step, else for each  $i_p \in \mathcal{T}$  do:

- (a) Let  $[j_1, \dots, j_p]$  be the sorted  $\{i_1, \dots, i_{p-1}, i_p\}$ .
- (b) Compute  $g = j_1 + j_2 n + j_3 n^2 + \dots + j_p n^{p-1}$ .
- (c) If  $g \notin \mathcal{V}_\tau$ , then add  $g$  to  $\mathcal{V}_\tau$ , else go to the next element in  $\mathcal{T}$ .
- (d) For each  $l \in \{i_1, \dots, i_{p-1}\}$  do:
  - i. Let  $[j_1, \dots, j_{p-1}]$  be  $\{i_1, \dots, i_{p-1}, i_p\} \setminus \{l\}$  sorted increasingly.
  - ii. If  $\mathcal{A}_{[j_1, \dots, j_{p-1}]} = \mathbf{false}$ , then set  $\mathcal{A}_{[j_1, \dots, j_{p-1}]} = \mathbf{true}$  and push  $[j_1, \dots, j_{p-1}]$  into  $\mathcal{Q}$ .

**Step 6.** If  $\mathcal{Q}$  is not empty, then go to **Step 3**, else stop.

**Output:**  $\mathcal{G}_\tau$ .

In **Step 1**,  $(\text{false}_n)^{p-1}$  is an  $n$ -dimensional vector of logical zeros, brought to power  $p - 1$  in the sense of the Cartesian product, and by that being a  $(p - 1)$ -dimensional logical square matrix  $n \times \cdots \times n$ . Indeed, here only one upper corner of this matrix is used, which includes only cells with non-repeating strictly decreasing subscripts. On the other hand, this matrix is memory demanding when  $p$  is high, in which case the matrix can be sparse. Then, some dynamic storing structure could be used instead, a search tree, say. **Step 2** aims at finding initial subscripts tuples. Usually, in **Step 2** (a), it takes only several attempts to find a proper  $u_1$ .

This algorithm is developed to directly search the  $\tau$ -th critical directions. It will be indicated in the experiments that much more *non-redundant* direction vectors can be eliminated from consideration than Paindaveine and Šiman’s procedure does. Furthermore, since we utilize a *ridge-by-ridge* search strategy, some tricks may be utilized to save considerable RAM; See **Step 5** for more details.

After obtaining  $\mathcal{G}_\tau$ , there are obvious ways to decode each  $g \in \mathcal{G}_\tau$  into  $\{i_1, \dots, i_{p-1}, i_p\}$  and compute the corresponding direction vectors. Based on these direction vectors, one may utilize the well-developed functions such as *qhull* (or *convhulln.m* in Matlab) (Barber *et al.*, 1996) to obtain all *vertices and facets* of  $\mathcal{D}_n(\tau)$  relying on the computed  $\mathcal{U}_\tau^*$  and the corresponding  $q_\tau^{(n)}(u_j)$ ’s.

## 4 Numerical performance

In this section, we investigate some data examples to illustrate the performance of the proposed algorithms. All of these results are obtained on a HP Pavilion dv7 Notebook PC with Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz, RAM 6.00GB, Windows 7 Home Premium and Matlab 7.8.

### 4.1 Real data

We start with a real data set. The data set is a part of the Blood Transfusion data set. It was first utilized by Yeh *et al.* (2009) and is available at <http://archive.ics.uci.edu>. The data set contains information of 748 blood donors randomly selected from the donor database of Blood Transfusion Service Center in Hsin-Chu City in Taiwan. It has been widely used to illustrate the performance of depth-based classifiers (Li *et al.*, 2012).

As the attribute M (total blood donated) is correlated with the attribute F (total number of donations), we have excluded it as did in Li *et al.* (2012)

as well. We analyze the subjects regarding three attributes:  $R$  (months since last donation),  $F$  (total number of donation), and  $T$  (months since first donation). After removing ties we got 501 three-dimensional observations. The scatter plot of this transformed data set is shown in Figure 3(a). We remark that the goal here is not to perform a thorough analysis of data, but rather to show how the algorithms work in practice.

We compute the critical direction vectors of seven depth regions of  $\tau = 0.025, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30$  by using the C++ implementations of the proposed algorithms, and then visualize them by **R**. The results are shown in Figures 3(b)-3(h). For each combination  $(n, \tau)$  of  $n \in \{125, 250, 501\}$  and  $\tau \in \{0.025, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$ , we report the  $\tau$ -th critical direction vectors obtained by the implementations of our two algorithms and that of Paindaveine & Šiman (2012b) for each depth region of the first  $n$ -th observations of this data set.

It turns out that both new algorithms result in the same number of direction vectors for each  $(n, \tau)$ , and the number of directions is always smaller than the upper bound  $n \times (n - 1)$  given in Corollary 1. Also, both yield precisely the same facets of the Tukey region. It is also found that the new approach yields the same results, namely, the same vertices and facets of the resulting convex body, as that (*coded in Matlab*) of Paindaveine & Šiman (2012b). However, in Paindaveine & Šiman (2012b) often many more directions are calculated: Table 1 reports the results in detail. Here  $M_{\text{new}}$  and  $M_{\text{PS}}$  denotes the numbers yielded by the two new algorithms and the method of Paindaveine & Šiman (2012b), respectively.

Further, we report the computation times in Table 1. Since the method of Paindaveine & Šiman (2012b) was implemented by Matlab, we also list the computation times of a Matlab implementation of Algorithm 2. Here  $T_{A1}^{\text{C++}}$ ,  $T_{A2}^{\text{C++}}$  correspond to the times of the C++ implementations of Algorithm 1 and Algorithm 2, respectively, and  $T_{A2}^{\text{Mat}}$ ,  $T_{\text{PS}}^{\text{Mat}}$  to the Matlab implementations of Algorithm 2 and the method of Paindaveine & Šiman (2012b), respectively. Table 1 shows that: (1)  $T_{A1}^{\text{C++}}$  remains almost the same for all  $\tau$  if  $n$  is given, while  $T_{A2}^{\text{C++}}$  is  $\tau$ -dependent as expected, (2) the implementation of Algorithm 2 runs much faster than that of Paindaveine & Šiman (2012b). (Note that there exists a threshold value for the iteration number in the implementation of Paindaveine & Šiman (2012b). Therefore, when  $n$  and/or  $p$  is larger, their procedure does not search all cones, and thus becomes faster relative to our procedure as indicated in Table 1. Our implementation has no such a threshold value.)

Table 1: The numbers of direction vectors and the computation times (in *seconds*) of the implementations of the proposed algorithms and that of Paidaveine & Šiman (2012b) for the Blood Transfusion data set.

$n$	$\tau$	Number of direction vectors			Computation times				
		$M_{\text{new}}$	$M_{\text{PS}}$	$M_{\text{PS}}/M_{\text{new}}$	$T_{A1}^{C++}$	$T_{A2}^{C++}$	$T_{A2}^{\text{Mat}}$	$T_{\text{PS}}^{\text{Mat}}$	$T_{\text{PS}}^{\text{Mat}}/T_{A2}^{\text{Mat}}$
125	0.025	258	2000	7.75	0.234	0.016	0.108	3.195	29.58
	0.05	604	4704	7.79	0.234	0.031	0.228	6.681	29.30
	0.10	1691	11904	7.04	0.219	0.078	0.646	15.680	24.27
	0.15	3131	22176	7.08	0.234	0.109	1.257	28.776	22.89
	0.20	4862	33760	6.94	0.234	0.171	2.160	44.148	20.44
	0.25	6203	48056	7.75	0.234	0.218	3.021	62.179	20.58
	0.30	7271	63240	8.70	0.249	0.234	3.881	84.055	21.66
250	0.025	1038	7504	7.23	1.825	0.125	0.473	10.372	21.93
	0.05	2808	18944	6.75	1.809	0.249	1.330	25.002	18.80
	0.10	8054	68368	8.49	1.826	0.702	5.230	94.901	18.15
	0.15	13455	120504	8.96	1.825	0.873	11.535	186.095	16.13
	0.20	18727	161608	8.63	1.840	1.168	22.171	237.267	10.70
	0.25	23553	200672	8.52	1.825	1.419	34.528	295.457	8.56
	0.30	27325	242224	8.86	1.872	1.669	46.187	360.738	7.81
501	0.025	3107	23656	7.61	16.178	0.664	2.355	35.339	15.01
	0.05	10903	81632	7.49	16.405	1.688	11.791	144.263	12.24
	0.10	31045	265912	8.57	17.942	4.473	72.788	455.988	6.26
	0.15	52578	453584	8.63	17.624	8.369	193.845	816.158	4.21
	0.20	74073	629120	8.49	18.221	9.420	442.025	1780.116	4.03
	0.25	94930	796072	8.39	16.683	11.383	605.090	1554.875	2.57
	0.30	110233	904800	8.21	16.252	13.397	810.595	1877.394	2.32

Table 2: Computation times (in *seconds*) of the implementation of Algorithm 1.

$p \backslash$	$n$							
	40	80	160	320	640	1280	2560	5120
3	0.0190	0.0790	0.5150	4.1722	34.3382	263.9043	2368.6762	20232.3972
4	0.1060	1.7311	28.1116	435.0931	7742.5234	-	-	-
5	0.9360	30.6073	1116.0143	-	-	-	-	-
6	6.8952	510.1846	-	-	-	-	-	-
7	40.4821	7198.3854	-	-	-	-	-	-
8	204.6081	-	-	-	-	-	-	-
9	879.8564	-	-	-	-	-	-	-

## 4.2 Simulated data

In the following, we run several simulations to investigate the performance of the proposed algorithms. The data are generated from  $p$ -dimensional standard normal distributions with  $p \in \{3, 4, \dots, 9\}$ , respectively. For each combination of  $n \in \{40, 80, 160, \dots, 5120\}$  and  $\tau \in \{0.01, 0.025, 0.05, 0.10,$



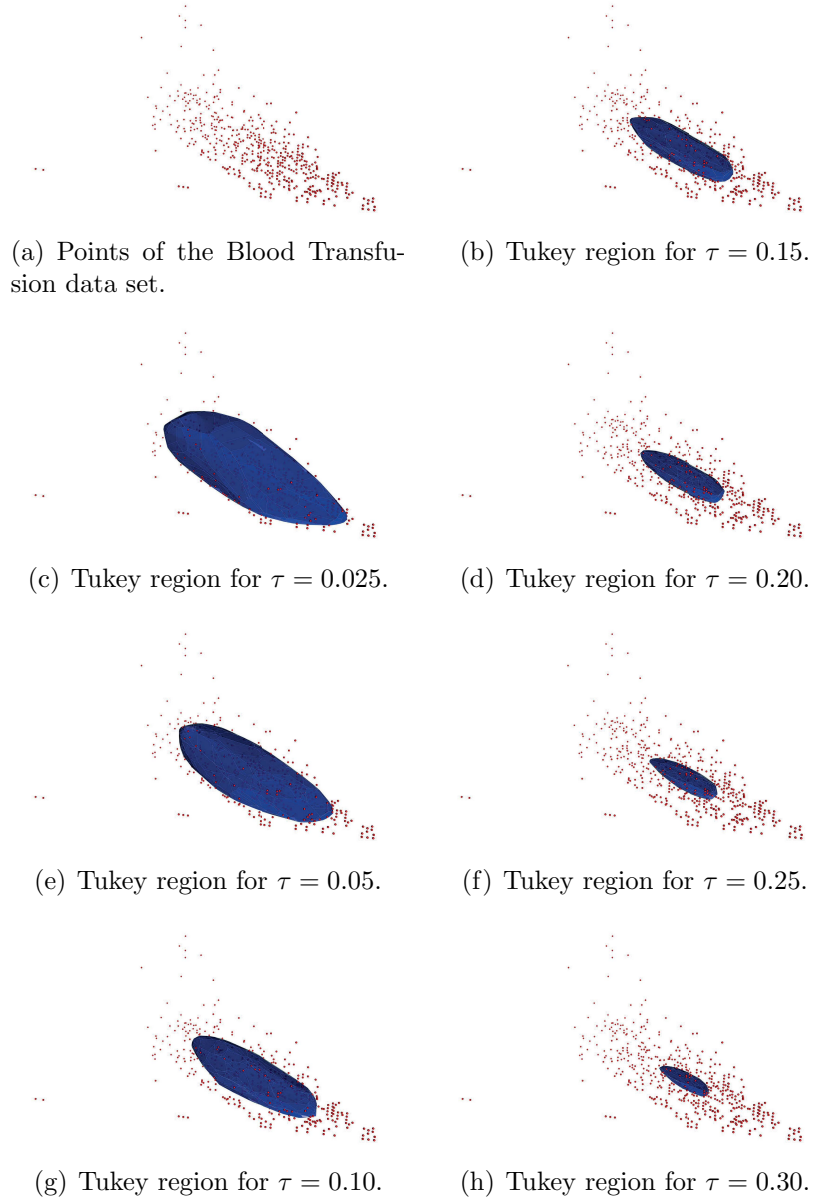


Figure 3: Shown are the scatter plot and the 0.025, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30-th Tukey depth regions of the Blood Transfusion data set.

0.15, 0.20, 0.25, 0.30}, we run the computation six times for each dimension  $p$ .

To grasp the dimensions of the solution, the average computation times of Algorithm 1 and Algorithm 2 are given in Table 2 and Table 3, respectively.

For the number of critical directions see Table 4. An additional experiment has shown that these critical directions coincide for the both algorithms in all considered cases.

Table 3: Computation times (in *seconds*) of the implementation of Algorithm 2.

$p$	$\tau \setminus$	Computation times							
		40	80	160	320	640	1280	2560	5120
3	0.010	0.0021	0.7720	1.0371	0.1290	0.5380	2.8392	17.6904	122.2731
	0.025	0.0050	0.0230	0.0500	0.2320	1.3961	8.0340	57.5669	480.6750
	0.050	0.0040	0.0150	0.0790	0.4600	3.1072	20.4828	165.1820	1345.7700
	0.100	0.0010	0.0780	0.1560	0.9828	7.8034	55.2943	497.0910	4913.0400
	0.150	0.0070	0.8720	0.2430	1.6641	12.7587	97.6757	858.5710	-
	0.200	0.0080	0.9758	1.3210	3.8412	19.2312	157.7640	1377.2864	-
	0.250	0.0650	1.0642	1.5840	6.0451	23.3143	201.8893	1710.5637	-
	0.300	0.8110	1.2680	1.4640	7.4832	27.3286	243.2335	-	-
4	0.010	0.2820	0.0120	1.0771	1.6811	5.8903	54.7433	883.2991	-
	0.025	0.0070	5.1943	1.5490	6.1093	40.4509	526.7044	-	-
	0.050	1.6120	4.0882	2.0321	12.1887	174.3240	-	-	-
	0.100	0.8424	2.2776	3.4320	49.6393	820.3323	-	-	-
	0.150	0.3370	2.5851	8.0374	116.2733	1883.5743	-	-	-
	0.200	1.3890	3.8851	12.5757	191.7885	3595.8481	-	-	-
	0.250	2.1030	4.2560	19.3621	300.2082	-	-	-	-
5	0.010	6.3383	1.0220	0.4680	8.1764	113.4221	-	-	-
	0.025	0.0312	0.2964	3.7908	49.4065	1094.6432	-	-	-
	0.050	0.4682	1.8861	18.5821	370.0534	-	-	-	-
	0.100	7.8468	8.8304	96.8406	2201.2864	-	-	-	-
	0.150	1.5170	9.3375	230.0543	6786.2751	-	-	-	-
	0.200	2.7290	17.6333	451.5117	-	-	-	-	-
	0.250	3.6490	24.8594	746.4839	-	-	-	-	-
6	0.010	0.9201	0.2270	9.2995	82.5747	-	-	-	-
	0.025	1.9188	1.8876	31.2469	1095.0842	-	-	-	-
	0.050	0.4330	8.0891	207.0145	-	-	-	-	-
	0.100	1.3572	43.1059	1537.9435	-	-	-	-	-
	0.150	2.8081	115.5934	-	-	-	-	-	-
	0.200	10.2566	263.8021	-	-	-	-	-	-
	0.250	17.8974	464.4488	-	-	-	-	-	-
7	0.010	0.8080	18.4181	25.9985	-	-	-	-	-
	0.025	2.8082	23.8524	435.0325	-	-	-	-	-
	0.050	4.7160	49.5613	-	-	-	-	-	-
	0.100	7.7224	360.0741	-	-	-	-	-	-
	0.150	24.6961	1259.3963	-	-	-	-	-	-
8	0.010	6.6553	20.6776	-	-	-	-	-	-
	0.025	7.2844	67.4749	-	-	-	-	-	-
	0.050	7.1604	373.2474	-	-	-	-	-	-
	0.100	37.1711	-	-	-	-	-	-	-
	0.150	84.6064	-	-	-	-	-	-	-
9	0.010	7.7512	39.1211	-	-	-	-	-	-
	0.025	14.7868	330.6523	-	-	-	-	-	-
	0.050	27.5246	-	-	-	-	-	-	-
	0.100	136.2527	-	-	-	-	-	-	-

Table 4: The number of direction vectors of the implementation of Algorithm 2.

$p$	$\tau \setminus$	Number of direction vectors							
		40	80	160	320	640	1280	2560	5120
3	0.010	20	34	148	454	1192	4752	17932	70628
	0.025	58	184	582	1620	6170	21412	82794	332502
	0.050	128	358	1258	4542	17442	63276	258140	1012602
	0.100	264	948	3292	12128	47748	183799	741100	2929610
	0.150	386	1399	5382	20176	81716	324704	1288903	-
	0.200	510	1896	7444	29224	118626	465519	1884020	-
	0.250	650	2482	9252	38524	148307	601866	2399212	-
	0.300	762	2816	11316	44734	177686	721658	-	-
4	0.010	122	148	795	4747	20783	120390	943866	-
	0.025	277	1178	4637	28020	177804	1232828	-	-
	0.050	697	2955	19095	112650	822341	-	-	-
	0.100	1661	10422	68561	514096	3944540	-	-	-
	0.150	3041	21351	143342	1157024	8918889	-	-	-
	0.200	4513	33054	242190	1898179	15375299	-	-	-
	0.250	6156	45064	370191	2859755	-	-	-	-
	0.300	762	2816	11316	44734	177686	721658	-	-
5	0.010	348	656	4228	41469	294280	-	-	-
	0.025	1238	6286	46926	354781	4212850	-	-	-
	0.050	3334	21585	224924	2528448	-	-	-	-
	0.100	10080	103176	1251496	17300876	-	-	-	-
	0.150	18054	246022	3318520	48699924	-	-	-	-
	0.200	32469	463202	6282398	-	-	-	-	-
	0.250	45838	687489	10352938	-	-	-	-	-
	0.300	762	2816	11316	44734	177686	721658	-	-
6	0.010	892	2651	22852	297244	-	-	-	-
	0.025	4432	32324	306260	4978120	-	-	-	-
	0.050	11797	145099	2164173	-	-	-	-	-
	0.100	44990	899011	18140464	-	-	-	-	-
	0.150	103296	2405588	-	-	-	-	-	-
	0.200	189524	4932541	-	-	-	-	-	-
	0.250	281686	8442180	-	-	-	-	-	-
	0.300	762	2816	11316	44734	177686	721658	-	-
7	0.010	2856	7994	125585	-	-	-	-	-
	0.025	13672	146756	2217514	-	-	-	-	-
	0.050	43318	751110	-	-	-	-	-	-
	0.100	175583	5706054	-	-	-	-	-	-
	0.150	476864	19113311	-	-	-	-	-	-
8	0.010	7287	29967	-	-	-	-	-	-
	0.025	40775	719030	-	-	-	-	-	-
	0.050	135444	3815565	-	-	-	-	-	-
	0.100	713671	-	-	-	-	-	-	-
	0.150	1908432	-	-	-	-	-	-	-
9	0.010	19610	104967	-	-	-	-	-	-
	0.025	114501	2602375	-	-	-	-	-	-
	0.050	429017	-	-	-	-	-	-	-
	0.100	2209657	-	-	-	-	-	-	-

## 5 Concluding remarks

Two new algorithms have been constructed for computing a  $\tau$ -trimmed Tukey region when  $p > 2$ . Our procedures relate to the HPS view on Tukey regions regarding direction cones as well as to the Kong-Mizera view regarding quantile envelopes. While the first algorithm comes without a particular search strategy, special search rules have been implemented in the second algorithm. They avoid unnecessary repeated checks and in turn save considerable computational times. The second algorithm, rather than searching the critical direction vectors cone-by-cone, finds all possible direction vectors ridge-by-ridge. The ridges are indexed by  $(p - 1)$ -tuples, which spares much RAM. Extensive data examples indicate that our results provide a significant speed-up over the HPS algorithm. The first algorithm, in passing through all observation hyperplanes, is obviously correct. The second algorithm is correct either. This is confirmed by our calculations, where in each and every case the second algorithm yields the same critical directions as the first one. Both algorithms have computational complexity  $O(n^p \log n)$ .

The literature contains many other depth notions, such as the projection depth and others, which, similar to the Tukey depth, have the projection property, that is, are equal to the minimum of univariate depths in any direction. It turns out that most of them can be computed by cutting convex polytopes with hyperplanes; see Mosler *et al.* (2009) and Liu & Zuo (2014) for details. By this, similar algorithms may be constructed to calculate the respective central regions in dimensions  $p > 2$ .

## Acknowledgements

Xiaohui Liu's research was supported by the National Natural Science Foundation of China (No.11461029), the Natural Science Foundation of Jiangxi Province (No.20142 BAB211014), and the Youth Science Fund Project of Jiangxi provincial education department (No. GJJ14350).

## References

- BARBER C.B., DOBKIN, D.P. AND HUHDANPAA, H. (1996). The Quickhull algorithm for convex hulls. *Computational Geometry* **46** 566–573.
- VAN BEVER, G. (2013). *Contributions to Nonparametric and Semiparametric Inference Based on Statistical Depth*. *PhD-thesis*. Université Libre de Bruxelles.

- DONOHU, D.L. (1982). *Breakdown Properties of Multivariate Location Estimators*. PhD-thesis. Harvard University.
- DYCKERHOFF, R. (2002). Inference based on data depth. In: Mosler, K., *Multivariate Dispersion, Central Regions and Depth: The Lift Zonoid Approach*, Springer, New York, (Chapter 5) 133–163.
- DYCKERHOFF, R. AND MOZHAROVSKIY, P. (2014). Exact computation of the halfspace depth. *Mimeo*.
- HALLIN, M., PAINDAVEINE, D. AND ŠIMAN, M. (2010). Multivariate quantiles and multiple-output regression quantiles: from  $L_1$ -optimization to halfspace depth. *The Annals of Statistics* **38** 635–669.
- KONG, L. AND MIZERA, I. (2012). Quantile tomography: using quantiles with multivariate data. *Statistica Sinica* **22** 1589–1610. Published online as [arXiv:0805.0056](#) [stat.ME] (2008).
- KONG, L. AND ZUO, Y. (2010). Smooth depth contours characterize the underlying distribution. *Journal of Multivariate Analysis* **101** 2222–2226.
- KOSHEVOY, G.A. (2002). The Tukey depth characterizes the atomic measure. *Journal of Multivariate Analysis* **83** 360–364.
- KOSHEVOY, G. AND MOSLER, K. (1997). Zonoid trimming for multivariate distributions. *The Annals of Statistics* **25** 1998–2017.
- LI, J., CUESTA-ALBERTOS, J.A. AND LIU, R.Y. (2012). DD-classifier: nonparametric classification procedure based on DD-plot. *Journal of the American Statistical Association* **107** 737–753.
- LIU, R.Y. (1990). On a notion of data depth based on random simplices. *Annals of Statistics* **18** 405–414.
- LIU, R.Y. (1992). Data depth and multivariate rank tests. In: Dodge, Y. (Ed.),  *$L_1$ -Statistical Analysis and Related Methods*, Elsevier, Amsterdam, 279–294.
- LIU, X. (2014). Fast implementation of the Tukey depth. *Mimeo*.
- LIU, X. AND ZUO, Y. (2014). Computing projection depth and its associated estimators. *Statistics and Computing* **24** 51–63.

- MILLER, K., RAMASWAMI, S., ROUSSEEUW, P., SELLARÈS, J.A., SOUVAINE, D., STREINU, I. AND STRUYF, A. (2003). Efficient computation of location depth contours by methods of computational geometry. *Statistics and Computing* **13** 153–162.
- MOSLER, K. (2013). Depth statistics. In: Becker, C., Fried, R., Kuhnt, S. (Eds.), *Robustness and Complex Data Structures: Festschrift in Honour of Ursula Gather*, Springer-Verlag, Berlin Heidelberg, 17–34.
- MOSLER, K., LANGE, T. AND BAZOVKIN, P. (2009). Computing zonoid trimmed regions of dimension  $d > 2$ . *Computational Statistics and Data Analysis* **53** 2500–2510.
- MOZHAROVSKIY, P. (2014) *Contributions to depth-based classification and computation of the Tukey depth*. PhD-thesis. University of Cologne.
- MOZHAROVSKIY, P., MOSLER, K. AND LANGE, T. (2014). Classifying real-world data with the  $DD\alpha$ -procedure. *Advances in Data Analysis and Classification*, to appear.
- PAINDAVEINE, D., ŠIMAN, M. (2011). On directional multiple-output quantile regression. *Journal of Multivariate Analysis* **102** 193–212.
- PAINDAVEINE, D. AND ŠIMAN, M. (2012a). Computing multiple-output regression quantile regions. *Computational Statistics and Data Analysis* **56** 840–853.
- PAINDAVEINE, D. AND ŠIMAN, M. (2012b). Computing multiple-output regression quantile regions from projection quantiles. *Computational Statistics* **27** 29–49.
- ROUSSEEUW, P.J. AND STRUYF, A. (1998). Computing location depth and regression depth in higher dimensions *Statistics and Computing* **8** 193–203.
- RUTS, I. AND ROUSSEEUW, P.J. (1996). Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis* **23** 153–168.
- SERFLING, R. (2006). Depth functions in nonparametric multivariate inference. In: Liu, R., Serfling, R., Souvaine, D. (Eds.), *DIMACS. Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, American Mathematical Society, Providence RI, 1–16.

- STRUYF, A. AND ROUSSEEUW, P.J. (1999). Halfspace depth and regression depth characterize the empirical distribution. *Journal of Multivariate Analysis* **69** 135–153.
- TUKEY, J.W. (1975). Mathematics and the picturing of data. In: James, R.D. (Ed.), *Proceeding of the International Congress of Mathematicians (Volume 2)*, Canadian Mathematical Congress, Vancouver, 523–531.
- YEH, A., SINGH, K. (1997). Balanced confidence regions based on Tukey’s depth and the bootstrap. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **59** 639–652.
- YEH, I.C., YANG, K.J., TING, T.M. (2009). Knowledge discovery on RFM model using Bernoulli sequence. *Expert Systems with Applications* **36** 5866–5871.
- ZUO, Y.J. (2003). Projection based depth functions and associated medians. *The Annals of Statistics* **31** 1460–1490.
- ZUO, Y.J. AND SERFLING, R. (2000). General notions of statistical depth function. *The Annals of Statistics* **28** 461–482.